

cats
Internet-Draft
Intended status: Informational
Expires: 27 August 2026

N. Tran
K. Nguyen-Trung
Y. Kim
Soongsil University
23 February 2026

Additional CATS requirements consideration for Service Segmentation-
related use cases
draft-dcn-cats-req-service-segmentation-03

Abstract

This document discusses possible additional CATS requirements when considering service segmentation in related CATS use cases such as AR-VR and Distributed AI Inference

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 27 August 2026.

Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	2
2. Terminology used in this draft	3
3. Example 1: AR-VR (XR) Rendering Sequential Subtask Segmentation	3
3.1. Expected CATS system flow	5
3.2. Impacts on CATS system design	5
4. Example 2: ML Model Vertical Partitioning Inference Parallel Subtask Segmentation	6
4.1. Expected CATS system flow	8
4.2. Impacts on CATS system design	8
5. Differences comparison between Normal and Service Segmentation CATS scenarios	9
6. CATS system design Consideration points to support Service Segmentation	9
7. Normative References	11
Authors' Addresses	12

1. Introduction

Service segmentation is a service deployment option that splits the service into smaller subtasks which can be executed in parallel or in sequence before the subtasks execution results are aggregated to serve the service request [draft-li-cats-task-segmentation-framework]. It is an interesting service deployment option that is widely considered to improve the performance of several services such as AR-VR or Distributed AI Inference which are also key CATS use cases [draft-ietf-cats-usecases-requirements].

For example, a recent 3GPP Technical Report on 6G use cases and services [TR-22870-3GPP] describes an XR rendering service that can be implemented as a sequential pipeline of subtasks, including a render engine, engine adaptation, and rendering acceleration. In contrast, an example of parallel service segmentation is parallel Machine Learning (ML) model partitioning for inference [SplitPlace], [Gillis]. Specifically, a ML model layer can be divided into multiple smaller partitions, which are executed in parallel. In both sequential and parallel segmentation cases, subtask may have multiple instances which are deployed across different computing sites.

This document analyzes these CATS service segmentation use case examples to discuss the impact of service segmentation deployment method on CATS system design.

2. Terminology used in this draft

This document re-uses the CATS component terminologies which has been defined in [draft-ietf-cats-framework]. Additional definitions related to service segmentation are:

Service subtask: An offering that performs only a partial functionality of the original service. The complete functionality of the original service is achieved by aggregating the results of all its divided service subtasks. Subtask result aggregation may be performed either in parallel or sequentially.

Service subtask instance: When a service is segmented into multiple service subtasks, each service subtask might have multiple instances that performs the same partial functionality of the original service.

3. Example 1: AR-VR (XR) Rendering Sequential Subtask Segmentation

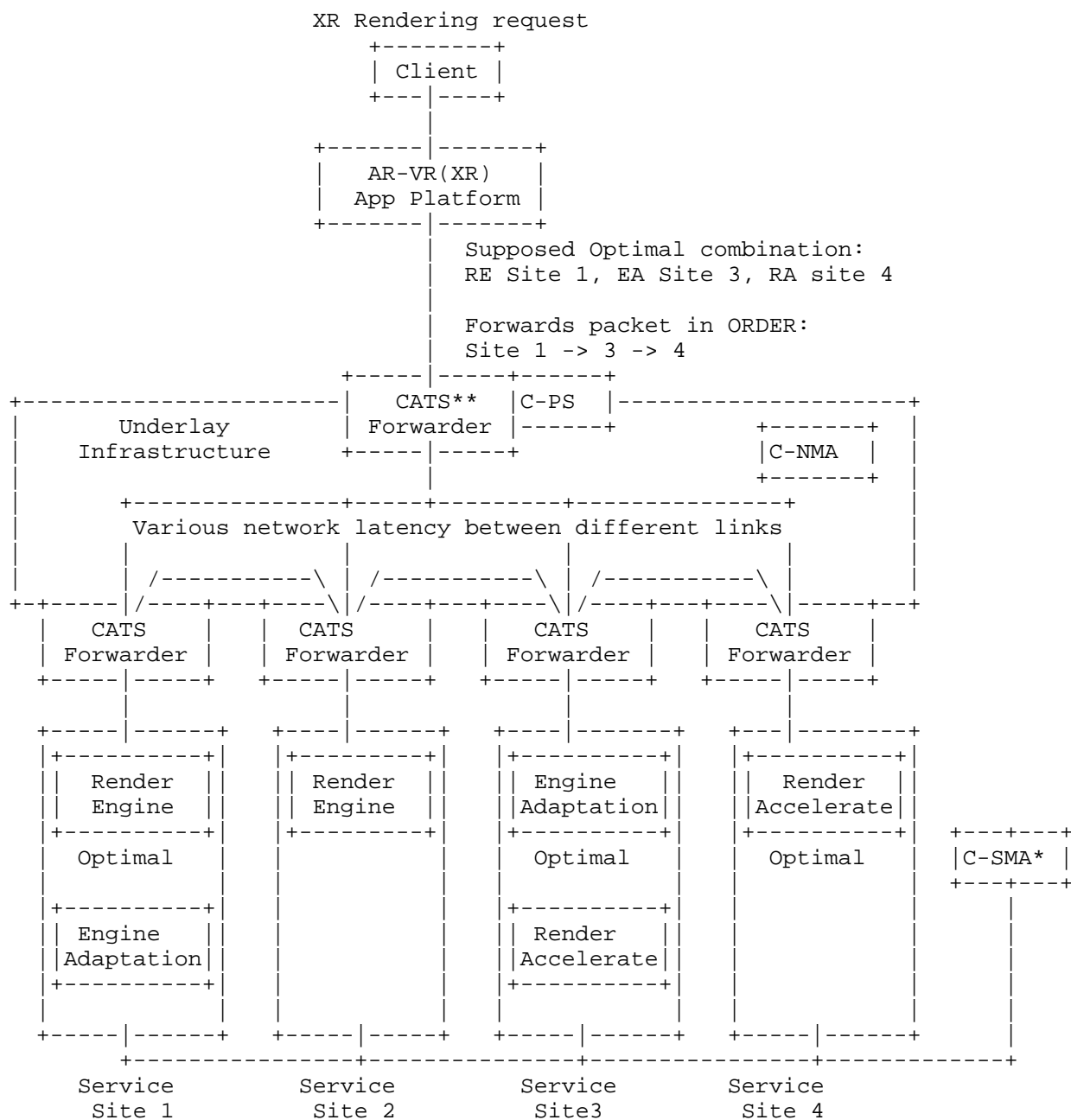


Figure 1: Example of a CATS system in Sequential Service Segmentation case

Figure 1 illustrates how a CATS system should perform optimal traffic steering for an XR rendering service deployed as a sequential pipeline of subtasks, including the render engine, engine adaptation, and rendering acceleration. This example is derived from the corresponding use case in [TR-22870-3GPP]. To return the rendered XR object to the client, the XR rendering request must be processed sequentially in the specified order by the three rendering subtasks.

3.1. Expected CATS system flow

- * The client sends an XR rendering request via its connected network to the XR application platform.
- * The XR application platform determines that the request should be processed by the XR rendering pipeline and forwards the packet via its attached CATS Forwarder.
- * The CATS Path Selector (CATS-PS) determines the optimal subtask composition of XR rendering pipeline and selects the most suitable instance for each subtask to steer the request. This selection is based on the current status of computing and network resources at the sites hosting the XR rendering subtask instances. For example, in Figure 1, the sequential pipeline consists of three subtasks: the optimal Render Engine instance is located at Site 1, the optimal Engine Adaptation instance at Site 3, and the optimal Rendering Acceleration instance at Site 4.
- * The CATS-PS configures the CATS Forwarder with routing information that specifies the required processing order: from Site 1 to Site 3 to Site 4.
- * The packet is steered through the CATS underlay infrastructure following the specified routing order and is sequentially processed at the designated service sites.
- * The XR application platforms returns the final processed XR rendering result to the client.

3.2. Impacts on CATS system design

- * A CATS system should provide a method to distinguish different CATS candidate paths corresponding to different service subtask instance combinations (different subtask composition or same subtask composition but different subtask instance location)
- * A CATS system should provide a method to deliver the service request to the determined optimal service subtask instance combination in correct order and correct composition.

4. Example 2: ML Model Vertical Partitioning Inference Parallel Subtask Segmentation

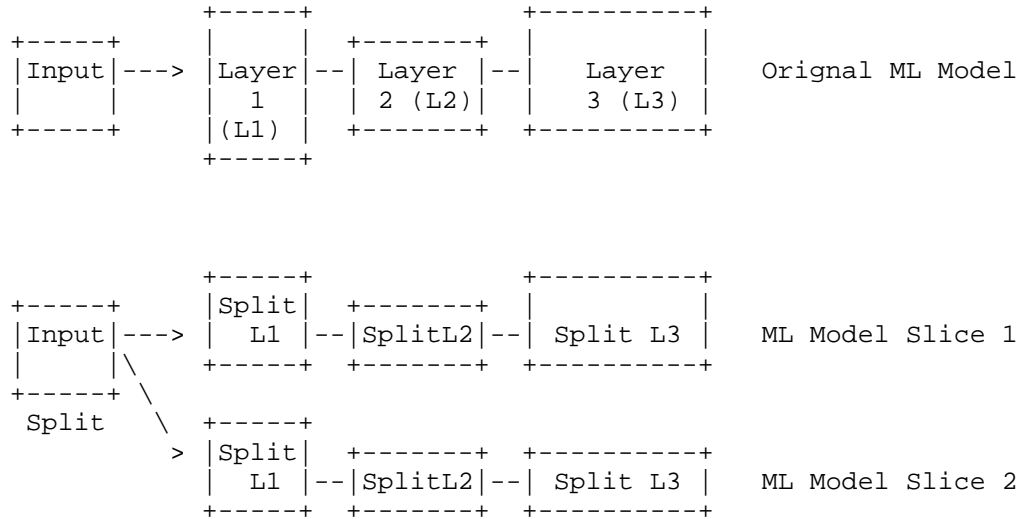


Figure 2: ML model Vertical Partitioning Illustration

Figure 3 illustrates how a CATS system can perform optimal traffic steering for a machine learning (ML) inference service deployed as a parallel pipeline of subtasks, where each subtask corresponds to a vertically partitioned slice of the original ML model. Based on the ML model splitting use cases described in [SplitPlace] and [Gillis], Figure 3 shows how an ML model can be vertically partitioned into slices that are executed in parallel to reduce inference response time. The input inference data from the client should be partitioned according to the input dimensions expected by each model slice. These slices then process their respective inputs in parallel, and the resulting outputs are merged to produce the final inference result, which is returned to the client.

4.1. Expected CATS system flow

- * The client sends an ML inference request via its connected network to the ML application platform.
- * The ML application platform determines that the request should be processed by the Vertical ML model partitioning pipeline.
- * The CATS-PS determines the optimal subtask composition for a vertically partitioned machine learning (ML) model pipeline and selects the most suitable instance for each subtask to steer the request. For example, in Figure 3, the ML model is partitioned into two vertical slices: Model Slice 1 is deployed at Service Site 1, and Model Slice 2 is deployed at Service Site 3.
- * The CATS Path Selector (CATS-PS) communicates its pipeline decision to the ML application platform. The platform then pre-processes the client's inference input into two smaller input slices, based on the input dimensions required by each model slice.
- * Each input slice is forwarded in parallel to its corresponding model slice instance at the designated locations (Site 1 and Site 3).
- * Once the processed outputs are returned from each model slice, the ML application platform merges them to produce the final ML inference result, which is then returned to the client.

4.2. Impacts on CATS system design

- * A CATS system should provide a method to distinguish different CATS candidate paths corresponding to different service subtask instance combinations (different subtask composition or same subtask composition but different subtask instance location)

- * A CATS system should coordinate with the segmented service platform entity to pre-process the original request data into the appropriate input formats required by the determined parallel subtasks.

5. Differences comparison between Normal and Service Segmentation CATS scenarios

In the normal CATS scenario:

- * The CATS system objective is selecting an optimal service instance to serve a service request
- * Different candidate CATS paths are caused by: service instances' computing and network resources status.
- * The CATS system delivers the service request to the determined optimal service instance

In the Service Segmenatation CATS scenario:

- * The CATS system objective is selecting an optimal service subtask combination. An optimal combination is composed of the optimal instances of each service subtask.
- * Different candidate CATS paths are caused by: service subtask instances' computing and network resources status, and possible different service segmentation variations (e.g. a service is segmented into different number of subtasks)
- * The CATS system delivers the service request to the determined optimal combination of service subtask instances in correct order (sequence/parallel) and subtask composition.

6. CATS system design Consideration points to support Service Segmentation

As AR/VR and Distributed AI Inference are among the CATS-supported use cases listed in [draft-ietf-cats-usecases-requirements], the CATS system should also fully support scenarios where service segmentation is applied to these use cases.

This section outlines three CATS system design considerations that are not yet addressed in existing CATS WG documents, including the Problem and Requirement document ([draft-ietf-cats-usecases-requirements]), the Framework document ([draft-ietf-cats-framework]), and the Metric Definition document ([draft-ietf-cats-usecases-requirements]):

- Traffic Steering Objective:
 - * The optimal service instance can be a sequence/parallel pipeline that consists the optimal instances of each subtask composing the service, instead of a single entity providing the complete service functionality, as assumed in the conventional CATS scenario.
- Traffic Steering Mechanism:
 - * The CATS system may be required to provide a mechanism to steer service requests in a predetermined sequence, as in the case of sequential service segmentation.
- CATS Metrics Aggregation:
 - * CATS metrics can be aggregated not only by metric category (e.g., computing, networking) but also by individual service subtasks. For instance, the CATS metric representing a candidate combination of subtasks may be derived by aggregating the metrics of its component subtasks.
 - * One possible realization of such metric aggregation is `_Service Pipeline Metrics_`.

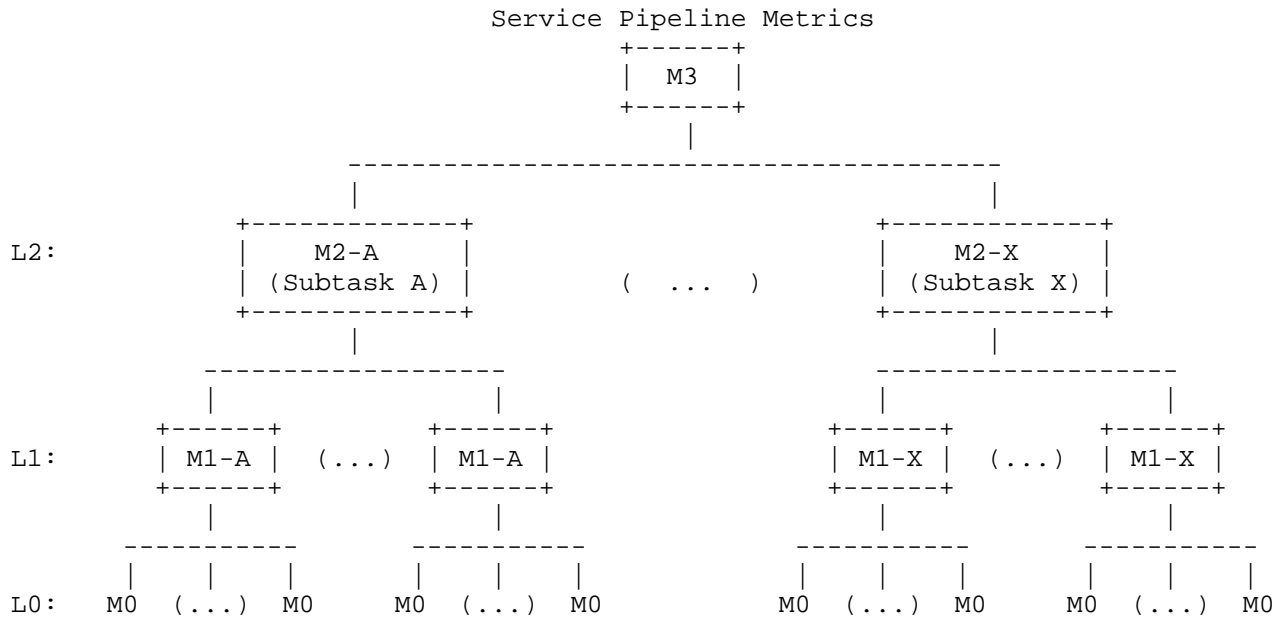


Figure 4: New CATS Metric Aggregation Level

The model organizes CATS-related measurements into multiple levels of abstraction. Level 0 (L0) metrics capture primitive measurements (e.g., resource, traffic, or system observations). Level 1 (L1) metrics are derived from L0 metrics to represent service-relevant aspects at a finer granularity. Level 2 (L2) metrics summarize the overall capability or suitability of each subtask (or subtask instance) by aggregating its L1 metrics. Finally, a pipeline-level metric (denoted as M3) is computed by aggregating the L2 metrics across all subtasks in a candidate pipeline, yielding a single score that can be used to compare candidate pipelines.

- * Aggregating metrics at the pipeline level can also reduce control-plane signaling overhead by avoiding the need to disseminate fine-grained metrics for each individual service replica. This consideration becomes increasingly important in large-scale 6G deployments, where the number of service replicas within a metro-area cell may grow to the order of hundreds.

7. Normative References

[draft-ietf-cats-framework]

Li, C., et al., "A Framework for Computing-Aware Traffic Steering (CATS)", draft-ietf-cats-framework, February 2026.

[draft-ietf-cats-metric-definition]

Yao, K., et al., "CATS Metrics Definition", draft-ietf-cats-metric-definition, February 2026.

[draft-ietf-cats-usecases-requirements]

Yao, K., et al., "Computing-Aware Traffic Steering (CATS) Problem Statement, Use Cases, and Requirements", draft-ietf-cats-usecases-requirements, February 2026.

[draft-ietf-spring-sr-service-programming]

Abdelsalam, A. , et al., "Service Programming with Segment Routing", draft-ietf-spring-sr-service-programming, February 2026.

[draft-lbddd-cats-dp-sr]

Li, C., et al., "Computing-Aware Traffic Steering (CATS) Using Segment Routing", draft-lbddd-cats-dp-sr, October 2025.

[draft-li-cats-task-segmentation-framework]

Li, C., et al., "A Task Segmentation Framework for Computing-Aware Traffic Steering", draft-li-cats-task-segmentation-framework, December 2024.

[Gillis] Yu, M., Jiang, Z., Chun Ng, H., Wang, W., Chen, R., and B. Li, "Gillis: Serving Large Neural Networks in Serverless Functions with Automatic Model Partitioning", October 2021, <<https://doi.org/10.1109/ICDCS51616.2021.00022>>.

[SplitPlace]

Tuli, S., Casale, G., and N. Jennings, "SplitPlace: AI Augmented Splitting and Placement of Large-Scale Neural Networks in Mobile Edge Environments", May 2022, <<https://doi.org/10.1109/TMC.2022.3177569>>.

[TR-22870-3GPP]

"Study on 6G Use Cases and Service Requirements", June 2025, <<https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=4374>>.

Authors' Addresses

Minh-Ngoc Tran
Soongsil University
369, Sangdo-ro, Dongjak-gu
Seoul
06978
Republic of Korea
Email: mipearlskal307@dcn.ssu.ac.kr

Kiem Nguyen Trung
Soongsil University
369, Sangdo-ro, Dongjak-gu
Seoul
06978
Republic of Korea
Email: kiemnt@dcn.ssu.ac.kr

Younghan Kim
Soongsil University
369, Sangdo-ro, Dongjak-gu
Seoul
06978
Republic of Korea

Phone: +82 10 2691 0904

Email: younghak@ssu.ac.kr